RasPi

DESIGN
BUILD
CODE

46

Get hands-on with your Raspberry Pi

XBOX ZERO ARCADE PART 1

+ PRINT WIRELESSLY FROM PI

Plus Add a battery pack to your Pi

Welcome

We're hoping that by the time this issue reaches you, the winter weather will finally be a thing of the past and warmer climes will be well underway. If that's not the case, though, we have the perfect project to complete indoors and out of the wind and rain. It's the first installment of a two-part series which shows you how to build a self-contained arcade machine out of old bits of kit, a spare Xbox pad and a Pi Zero! You'll need to come back next month to discover how to load it with your favourite retro games.

Other highlights this month include a handy way to power up your Pi by adding a battery pack and a way to print wirelessly to any printer on your network.

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

Editor

From the makers of

LinuxUser & Developer

Join the conversation at...

@linuxusermag | Linux User & Developer | linuxuser@futurenet.com

Contents

Xbox Zero arcade Pt 1

Let's make a self-contained arcade machine out of old bits of kit, a spare Xbox pad and a Pi Zero!

The Raspberry Pi Zero is tiny, ridiculously tiny. It's also small enough to be hidden in a variety of household objects in order to enhance their capabilities. Whatever you can find to fit it in, you can turn into some kind of smart machine.

Take old game controllers. If you're anything like us you've probably got a couple of boxes full of old computer equipment you just can't bear to throw away – an Atari Jaguar that hasn't been touched since the 90s, a Sega Dreamcast which you're sure you'll plug in again one day, an old Xbox that lies languishing since you picked up something bigger and better. Turns out it actually was useful to keep them around – it's time to bring these old systems back to life.

We're going to show you how to gut an old videogames controller, replace its innards with a Raspberry Pi Zero, and then load it up with a treasure trove of retro games. From start to finish, this project should take you under an hour to complete – and then you'll be able to load up the ROMs you legally own on your new console and enjoy them from the comfort of your sofa.

THE PROJECT ESSENTIALS

- **Raspberry Pi Zero**
- **Original Xbox controller**
- **Wire cutters**
- **Craft knife**
- **Isopropyl alcohol swabs**
- **Micro SD card**
- **BluTak**
- **Micro USB OTG cable**
- **Cross-head screwdriver**
- **Electrical tape**
- **2A micro USB power supply**

01 Gather your equipment

While the Zero doesn't take up much space, videogame controllers are often stuffed full of delicate electronics. The trick here is to find a games controller which has enough space inside for the Zero. We're going to be using the original Xbox controller, nicknamed The Duke. If you don't have one to hand, they can be picked up for a couple of quid from most second-hand electronics shops, and they're easily found online too.

If you can't find one, you can use newer USB game pads that are designed to look like controllers for classic systems like the SNES and Mega Drive. Make sure you choose a controller that has enough buttons for the games you want

to play – some classic fighting games, for example, really can't be played on a two-button NES controller.

02 Warning!

Working with electrical items and sharp objects can be dangerous. You risk damaging yourself or, worse, breaking your toys. Please ensure everything is unplugged from electrical supplies before attempting this project. As with any electronics projects, you should also take care to fully ground yourself before playing around with sensitive components –the static electricity from your body can ruin them. Anti-static wrist straps or a few taps on a radiator should do the trick.

Below You can mod your controller with just a few simple tools

03 The build

You should be now have a reasonably good idea of the controller that we'll be working with. 'The Duke' has dual joysticks, six buttons, a D-Pad and two triggers – and it's compatible with most retro games systems.

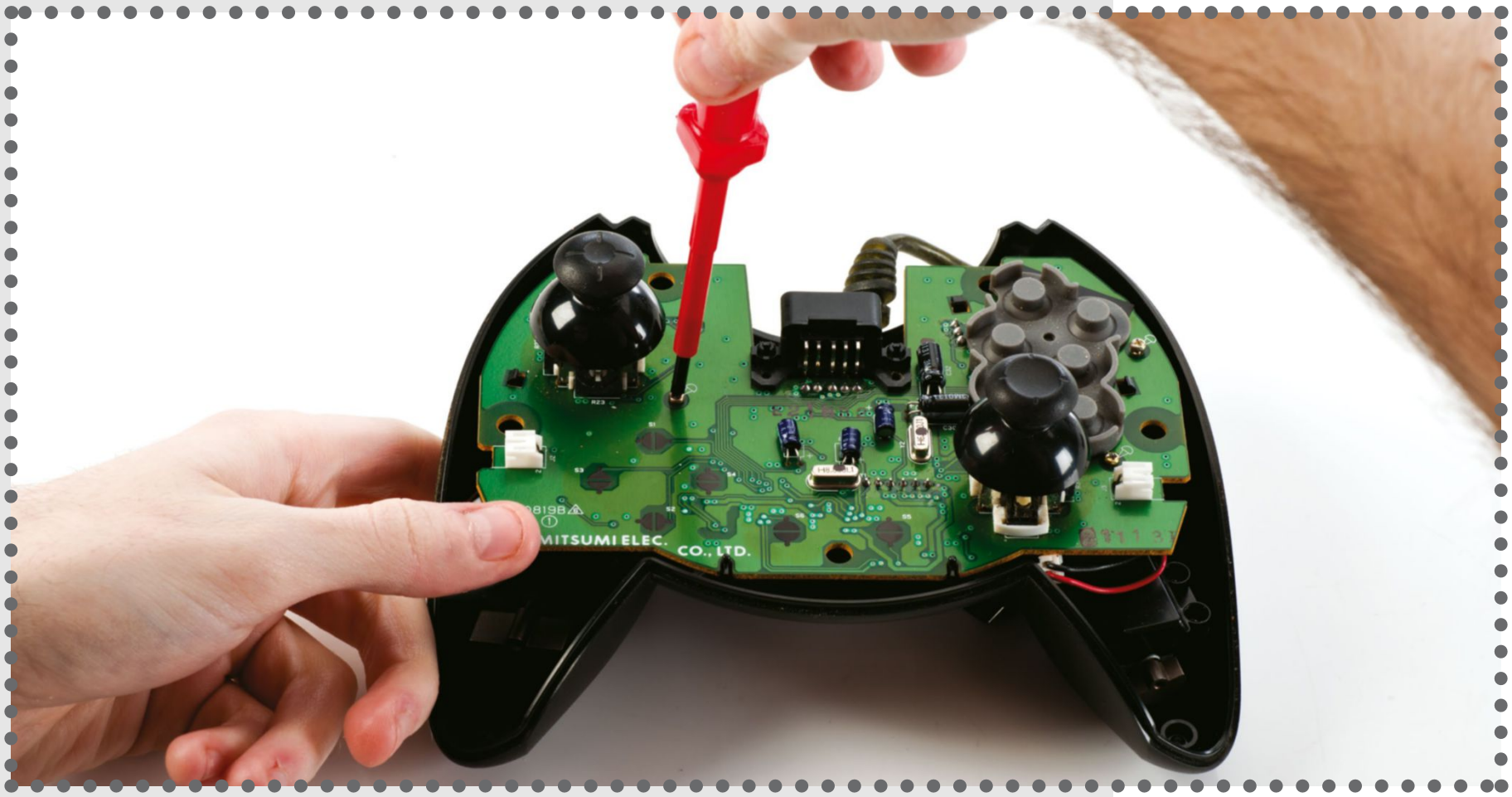
04 Fitting

If you're using a different controller, double-check that the Pi is likely to fit inside before you crack it open. As you'll see here, the Pi nestles neatly between the triggers of this controller – the original Xbox controller is one of the largest.

05 Unscrewing

The controller is held together by half a dozen cross-head screws. Be careful when opening the case as the buttons and rubber contacts are loose within the controller – they will spill everywhere!

Below Be careful that you don't lose any small parts when opening up the controller



06 Opening

With the shell removed, you should be able to undo the screws holding the main circuit board in place. There are also a couple of connectors which power the vibration motors – gently unclip them in order to completely remove the board. You might find it easier to use a pair of pliers for this – just be very gentle as you pull!.

07 Gently does it

You can see for yourself just how well the Pi fits here; it can be squeezed under the memory card slot. If you want to hold it firmly in place, use some BluTak as a temporary solution. Also, if you're using an older controller, it's worth giving it a bit of a clean. Remove the rubber contacts and gently swab under them using the isopropyl alcohol swabs.

08 Cut to fit

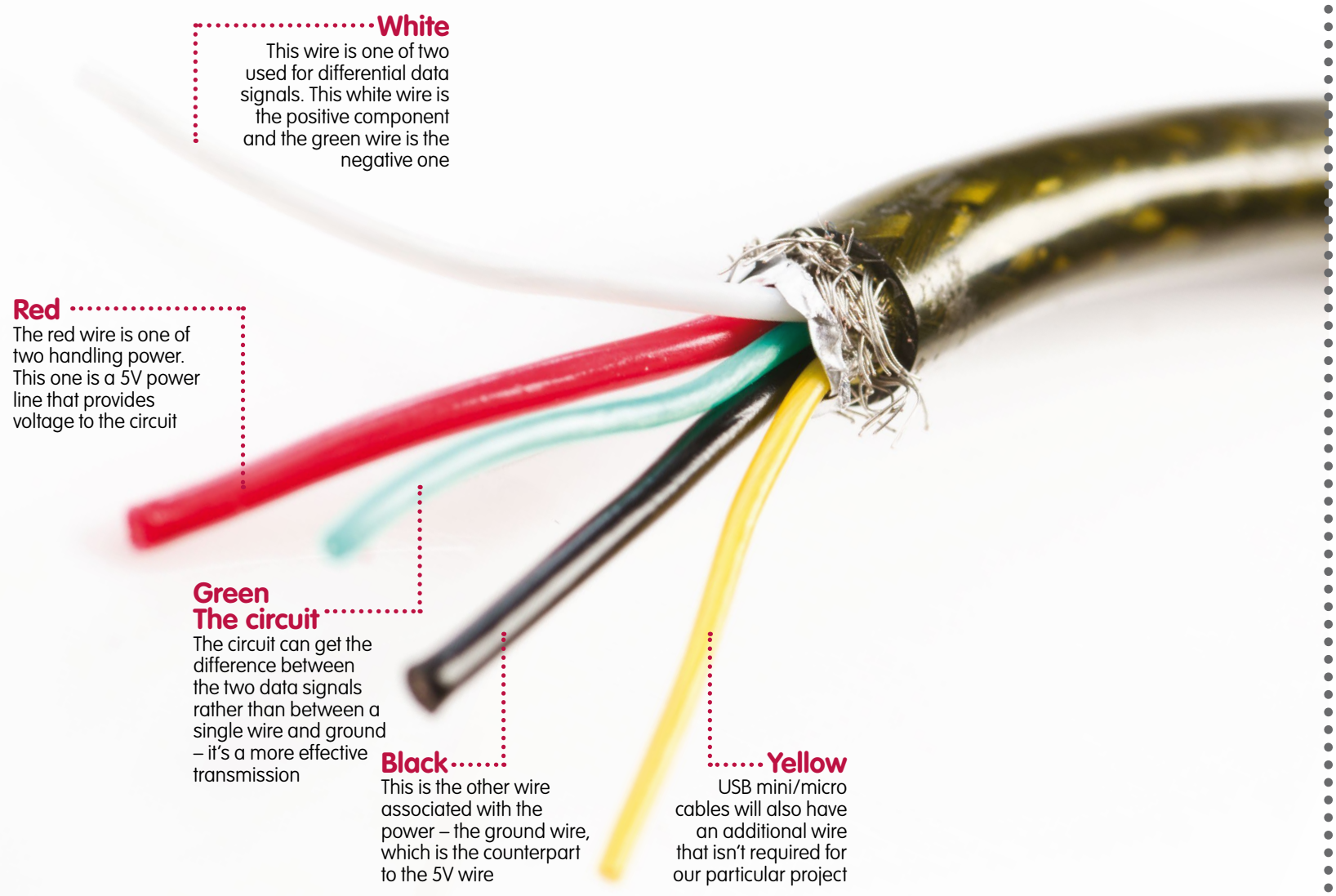
Depending on the model of controller, you may find that the Pi blocks one of the internal plastic struts. The plastic is soft enough that a craft knife will easily cut it down to size, though. Start with small strokes, shaving off a tiny bit at a time until you have enough room. Make sure the plastic dust is cleaned out before you reassemble the controller. If you have a can of compressed air, you can use it to easily blow away the shavings.

09 Connecting it up

If you're using a controller that has a regular USB port on it, you can just plug it into the Pi via a USB OTG converter. If you're using the original Xbox Controller, it's slightly tricky. Microsoft, in its infinite wisdom, has decided that the original Xbox should use USB – but with an

Lots of little bits

When taking apart electronics, keep a few small bowls or containers nearby. Put each type of screw in its own separate container so you don't accidentally mix them up. With the Xbox controllers, you'll find that the buttons especially have a habit of rolling away from you, so stash them somewhere safe as well. Keep track of any random bits of plastic or rubber which may be useful in re-assembling.



incompatible plug design. This means, in order to connect the controller to the Pi, we need to do some wire stripping. Fun!

The wiring inside the Xbox controller's cable uses bog-standard USB wiring colours, so once you've chopped the plugs off the controller and the OTG cable, it's pretty straightforward to connect them together.

10 Wiring

Strip the wires by a couple of centimetres and then connect them together. You should have Red, Green, White, and Black. The Xbox cable also has a Yellow wire which you can ignore. It is worth noting at this point that you need to be sure that you have a USB data transfer cable and not just a plain old power cable – the former will look like the

photo above, but power cables will be missing the two data wires.

With the wires stripped, we temporarily used regular sticky-tape to make the connections between the OTG cable and the controller – for a more permanent installation, you can use electrical tape or simply solder the wires together.

11 Insulation

One thing to note: you'll need to insulate the bottom of the Pi against all the contacts on the controller. For this quick hack, we've used some of the cardboard packaging – but any non-conductive material will do.

From there, it's as simple as screwing the case back together. Make sure that the controller's buttons and joysticks don't slip out of alignment. Keep track of which coloured buttons go where and you should be fine.

12 Wiring up

The Pi will need three wires connected to it in order to work. The controller cable needs to be connected to the USB OTG port. An HDMI cable goes from your TV to the mini HDMI port on the Pi. Finally, a 2A micro USB power supply needs to be plugged into the Pi's power socket. We've used a standard mobile phone charger, but you can use a USB

Left You can solder the OTG cable and controller together, but sticky-tape will also do the trick

The right controller

Second-hand stores like CEX or GAME often have some older, obsolete consoles and accessories out of public view, as they aren't particularly high-selling these days. It's worth asking the staff what they have if you can't see what you need on display. Some charity shops also have old consoles for sale. Failing that, local car boot sales or simply asking your gamer friends are both excellent ways to grab inexpensive controllers for all sorts of consoles.

battery pack if you want to reduce the number of wires trailing around your room.

13 A word about power

You might be wondering whether it's possible to get the HDMI cable to supply power from the TV to the controller. Sadly, the HDMI specification doesn't permit power to flow in that direction. If your TV has a USB socket on it, you could use that to supply the Pi with power – just make sure the socket itself is powerful enough. The Pi needs at least 1 Amp, and ideally 2 Amps. Many TVs will only output 500mA which isn't enough to run the Pi.

14 Let's play!

Okay, It's looking good – you're nearly ready to play and next issue we'll show you how to get some games onto your new device using some emulation software.

"If your TV has a USB socket, you could use that to supply the Pi with power – just make sure it's powerful enough"

Picture perfect Pi

Using a Pi to infuse AI into a camera and shock users into taking beautiful photographs

From alarm-clocks and to-do lists to calendar notifications and email reminders, do you sometimes get the feeling that you're a slave to the machines?
Peter Buczkowski has just taken the slavery to the next level. His Prosthetic Photographer 'looks' through a digital camera for interesting scenes and when it finds one, it jolts you with an electric shock, forcing your index finger to involuntarily trigger the camera's shutter and snap the image. Ouch… lovely.

What was the original inspiration behind the Prosthetic Photographer project?
Prosthetic Photographer is part of my Master's thesis in digital media. The topic I chose is 'Experiments on human-computer interaction through electrical body part stimulation'. I discovered TENS units (transcutaneous electrical nerve stimulation) that people usually use for pain relief. One can also use them to stimulate specific nerves and thus move a muscle unwillingly. I really liked the idea of having a way to control human behaviour with code, and to create a new form of human-computer interaction where the human becomes the interface. For this project I wanted to get some insight into machine learning and neural networks and how to use them for creative work. I wanted to create a device that knew about 'good- looking' and aesthetic images, and which controls the human using it to take them – and eventually even [have them] learn from the decisions the camera had made.

Peter Buczkowski
Peter is a designer and creative technologist from Bremen in Germany who works on projects inspired by different areas of the digital world.

How do you train the system to judge whether what it's looking at is click-worthy?

I used a dataset called CUHK-PhotoQuality which consists of around 17,000 images (http://mmlab.ie.cuhk.edu.hk/archive/CUHKPQ/Dataset.htm). These were submitted and labelled by photographic communities online; they've also been categorised into high and low-quality images. Transfer learning was used as the training method, which relies on using an already trained neural network with just its last layer re-trained with one's desired dataset. I used Google's Inception Model, which is a neural network that specialises in image classification. During the training process, which was done for 4,000 iterations, 80 per cent of the dataset was used to train the network to be able to classify between the given two categories of image quality. The result was tested against the remaining 20 per cent and achieved a training accuracy of over 90 per cent. This meant that

Left The project uses Google's Inception Model neural network. It was trained on the CUHK-PQ dataset that consists of 17,613 images obtained from a variety of online communities, divided into seven semantic categories and labelled as high and low quality. The pre-trained weights were then transferred to the Pi.

new images could be categorised with very high precision. Later I decided to let the system trigger a photo only when the image in front of it is seen as at least 95 per cent 'high quality'.

What was the most challenging part of the project?
Getting around the whole terminology of machine learning and neural networks. I have still only just slightly scratched the surface of the whole topic, but managed to reach my goal in the end. The computational power was also a challenge – training the network, even though I used transfer learning as a method, takes a lot of time. One challenge that I had given myself was to create a method to use the project without any stick-on electrodes – most projects to do with electrical impulses use those. You have to apply them to the body and they can't be reused after they've been on a for a while. My electrodes are placed on the project itself using aluminium tape; everybody can just grab the project and use it.

Left Peter used the TNS SM 2 MF TENS unit (http://bit.ly/lud_tens) which can turn up the intensity of the shock to 75mA. While half-power was enough for Peter to use the system, other people had to turn it up to full, or even amplify the signal using special electrode gel like the one available through Amazon, http://bit.ly/lud_gel.

Any particular reason for using the Raspberry Pi?
My main reason was that I wanted to make the system mobile. For that I needed a powerful computer that can also collect image data. I was familiar with the Pi from other projects, and the combination of the Picamera module and the Raspberry Pi 3 was perfect for this project. The Picamera was super-convenient to use and gave better results than I expected. Also, the Pi is powerful enough to run the image classifier every four seconds, which again was way faster than I expected.

Do you plan to extend the project?
I'm thinking about a commercial version that would include just the upper parts to always show the current quality of the scene, somewhat similarly to a light meter. I also want to explore using different databases. One idea, for example, was to just use images from Instagram that had a lot of likes. This could result in a complete different aesthetic; my current system really enjoys the colour blue because of all the nature images in the dataset! It would also be interesting to have enough computing power to run the system in real time, so that you don't have to point the camera at a scene for four seconds to see if it's worth photographing.

Like it?

Peter completed his Masters in Arts in the Digital Media program of the University of the Arts in Bremen in 2017. You can find his other projects in his portfolio at **http://peterbuczkowski.com**

Further reading

Peter's projects range from representations/interpretations of theoretical topics over functional products, to applied software. He now wants to focus on computer game-connected projects, such as his previous projects Current Times and Twitch.


POWER
SONY
E-mount

Print wirelessly with your Pi

Breathe new life into an old printer by using your Raspberry Pi as a wireless print server

Wireless printing has made it possible to print to devices stored in cupboards, sheds and remote rooms. You don't have to own a shiny new printer for this to work; old printers without native wireless support don't have to end up in the bin, thanks to the Raspberry Pi.

The setup is simple. With your Pi set up with a wireless USB dongle, you connect your printer to a spare USB port on the computer. With Samba and CUPS (Common Unix Printing System) installed on the Raspberry Pi, all that is left to do is connect to the wireless printer from your desktop computer, install the appropriate driver and start printing.

CUPS gives the Raspberry Pi a browser-based admin screen that can be viewed from any device on your network, enabling complete control over your wireless network printer.

THE PROJECT ESSENTIALS

Latest Raspbian image
USB printer
USB wireless card

01 Check your printer works

Before starting, check that the printer you're planning to use for the project still works and has enough ink. The easiest way to do this is to check the documentation (online if you can't find the manual) and run a test print.

02 Detect your printer

With your Raspberry Pi set up as usual and the printer connected to a spare USB port, enter:

```
lsusb
```

This will confirm that the printer has been detected by your Raspberry Pi. In most cases you should see the manufacturer and model displayed.

03 Install Samba and CUPS

Install Samba on your Pi to enable file and print sharing across the entire network:

```
sudo apt-get install samba
```

Next, install CUPS:

```
sudo apt-get install cups
```

With a print server created, begin configuration by adding default user 'pi' to the printer admin group:

```
sudo usermod -a -G lpadmin pi
```

04 Set up print admin

Set up the CUPS print admin tool. Boot into the GUI (startx) and launch the browser, entering 127.0.0.1:631. Switch to Administration, before ensuring that the 'Share printers' and 'Allow remote administration' boxes are selected. Select Add Printer and proceed to enter your Raspbian username and password.

05 Add your printer

A list of printers will be displayed, so select yours to proceed to the next screen where you can confirm the details, add a name and check the Share This Printer box. Click Continue to load the list of printer drivers and select the appropriate one from the list.

06 Configure Samba for network printing

Using a Windows computer for printing? Samba will need some configuration. Open '/etc/samba/smb.conf' in nano, search (Ctrl+W) for '[printers]' and find 'guest ok' which you should change as follows:

```
guest ok = yes
```

Next, search for "[print$]." Then change the path as follows:

```
path = /usr/share/cups/drivers
```

07 Join a Windows workgroup

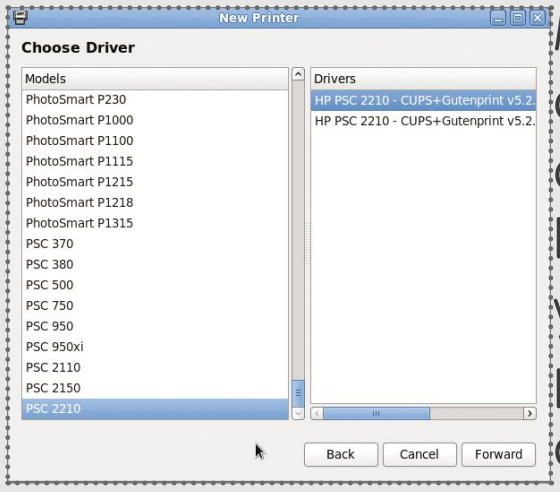
With these additions made, search for "workgroup" in the configuration file and then add your workgroup:

```
workgroup = your_workgroup_name
wins support = yes
```

Make sure you uncomment the second setting so that the print server can be seen from Windows. Next, save your changes and then restart Samba:

```
sudo /etc/init.d/samba restart
```

08 Accessing your printer



Meanwhile, it's a lot easier to access your wireless printer from a Linux, Mac OS X or other Unix-like system, thanks to CUPS. All you need to do is add a network printer in the usual way and the device will be displayed.

09 Add AirPrint compatibility

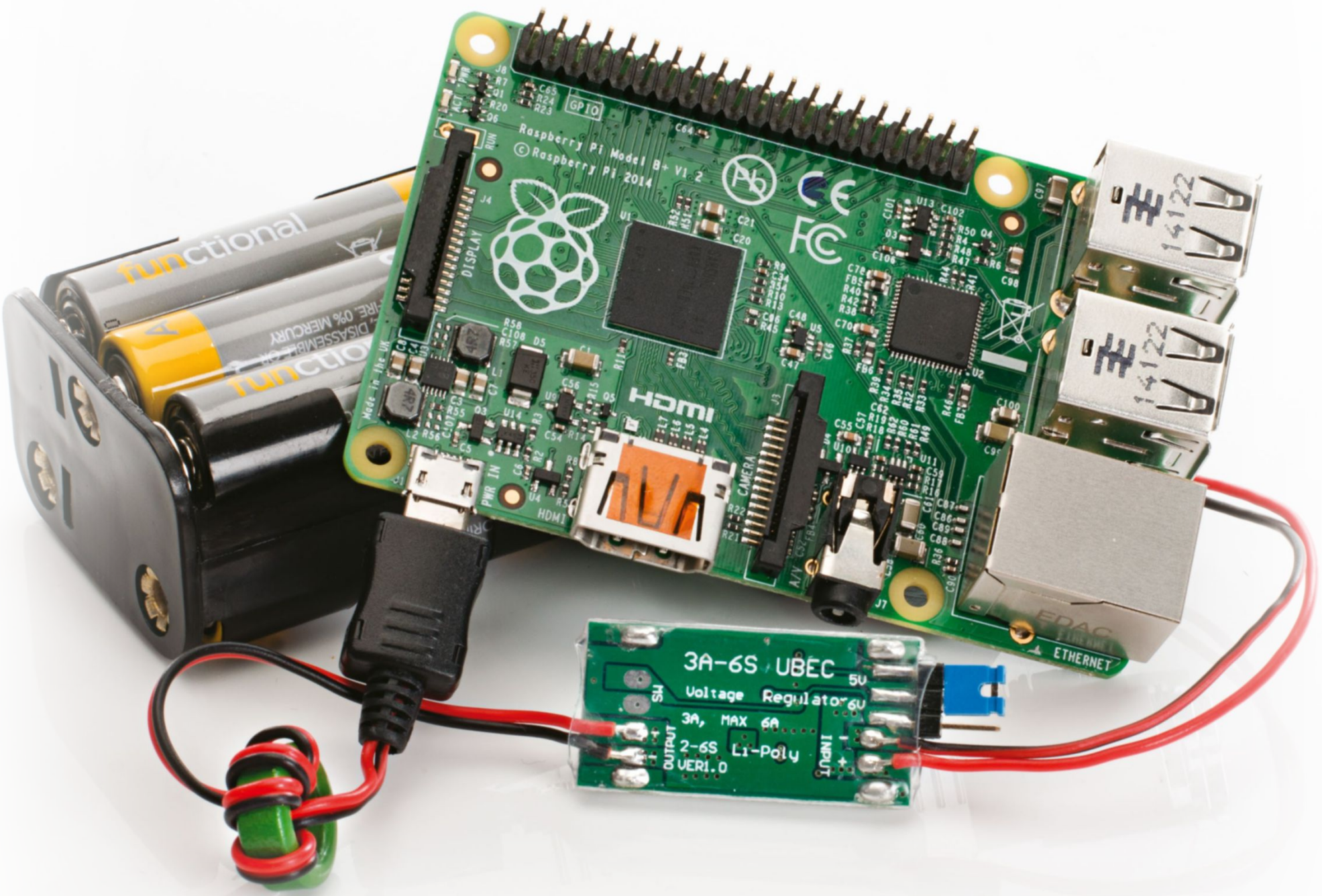
It's also possible to print wirelessly from your Apple iPad using Apple's AirPrint system. To do this, you need to add the Avahi Discover software:

```
sudo apt-get install avahi-discover
```

Your wireless printer will now be discoverable from your iPad or iPhone and will be ready to print.

Add a battery pack to your Raspberry Pi

Don't leave your Raspberry Pi behind – incorporate it into mobile projects by powering it up using humble AA batteries



Your Raspberry Pi's mobility is usually restricted by the length of the power lead. Rather than limiting it to your desk or living room, however, you can use it for mobile projects as diverse as launching it into near-Earth orbit or monitoring and automating your garden.

Of course, to do this you will need batteries, but adding battery power to your Raspberry Pi is simpler than you might have imagined. All that is required are six rechargeable AA batteries (or single-charge alkaline), a battery box with space for the batteries and a UBEC. The latter is a Universal Battery Elimination Circuit, a voltage regulator that will regulate the power supply and prevent damage to the Raspberry Pi, and can be bought for under £10.

01 Make your order

If you're buying your components online, you should be able to get them all within five days. However, if you're ordering offline (specifically the UBEC), you should avoid traditional electronics stores and instead visit a model enthusiast store, as these circuits are regularly used in RC devices.

02 Check your UBEC

Two types of UBEC are available to choose from. If you used the store that we suggest in the project essentials box to the left, you'll receive one with a micro USB power connector for easy connection to your Raspberry Pi. However, if you bought one from eBay then there is a strong chance that you will receive one with a 3-pin connector.

THE PROJECT ESSENTIALS

AA battery box
bit.ly/1FDiJGa
3-Amp UBEC
bit.ly/1HLKih7
3-Amp terminal strip
6x AA rechargeable batteries

03 Move connector pins

In order to use the UBEC with a 3-pin connector, you'll need to alter the position of the pins so that they occupy the two outer slots. Just use a small jeweller's screwdriver to lever up the small plastic catch and remove the red wire from the central slot, before sliding into the unoccupied outer slot.

04 Connect to battery box

With five batteries in the battery box, connect it to the UBEC (red-to-red, black-to-black) by twisting the wires, soldering or employing a 3-amp terminal strip, cut down to two pairs. It can be cut to size using a modelling knife.

05 Add a battery to boot

With your Pi ready to use and your Wi-Fi dongle plugged in, connect the UBEC to the micro USB port and insert the sixth battery into the battery box. The Pi's power and status lights should indicate that the computer is booting up, which gives you a fully portable computer.

06 Connect the 3-pin UBEC

If you purchased the UBEC with the now-modified 3-pin connector, you'll need to connect this to the Raspberry Pi's GPIO header. Connect the positive +5V (red) connector to Pin 2 and the negative 0V connector to Pin 6.

07 Measure uptime

You should have already set up your Pi for SSH use, so connect to the device via Putty after giving it time to boot fully (at least 60 seconds). In the terminal, enter:

```
sudo dd bs=32m if=/Users/ rachelcrabb/
```

```
Desktop/ArchLinux/  archlinux-hf-2013-02-11.
img      of=/dev/disk1
```

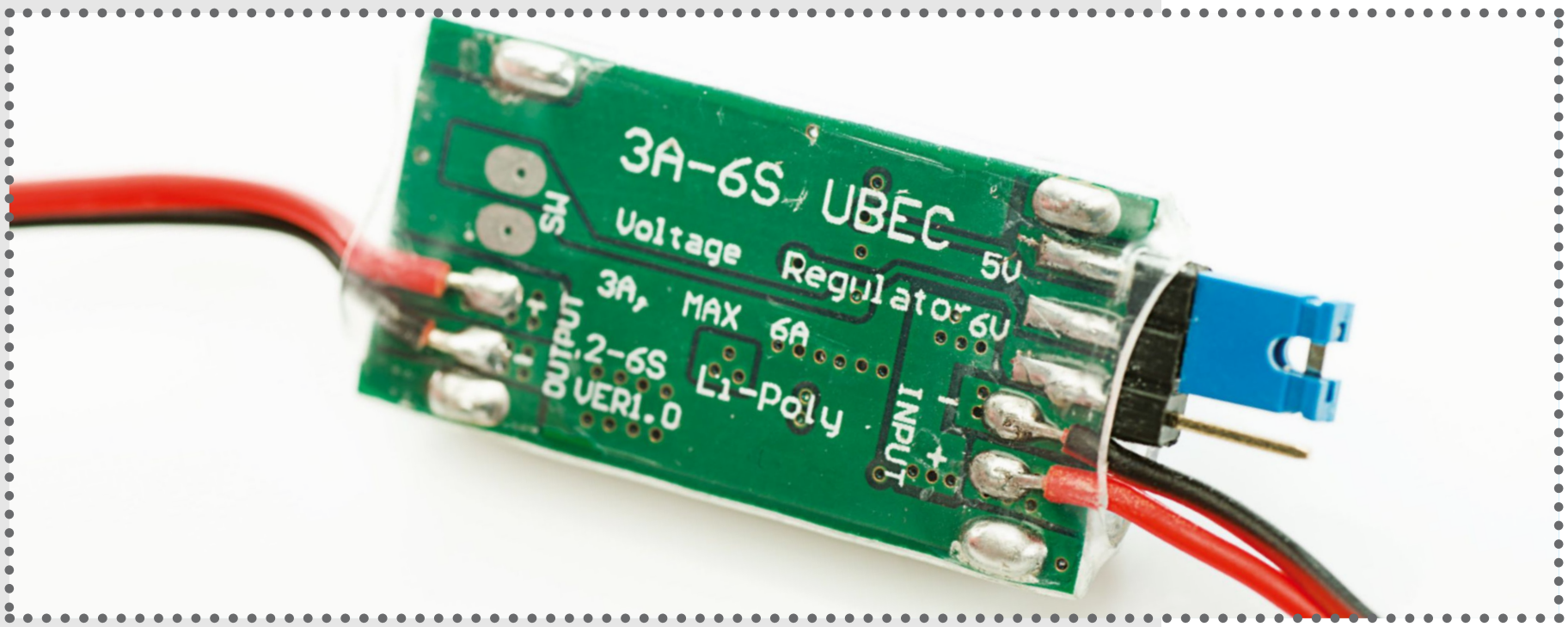
This command will display the system uptime and also keep the Wi-Fi connection active.

08 Judge your uptime results

Uptime results depend upon the type of battery you use and the Raspberry Pi model. Single-charge batteries will last a little bit longer, but this is a more expensive option. Meanwhile, newer models have greater power requirements but run for less time. For more power, add more batteries!

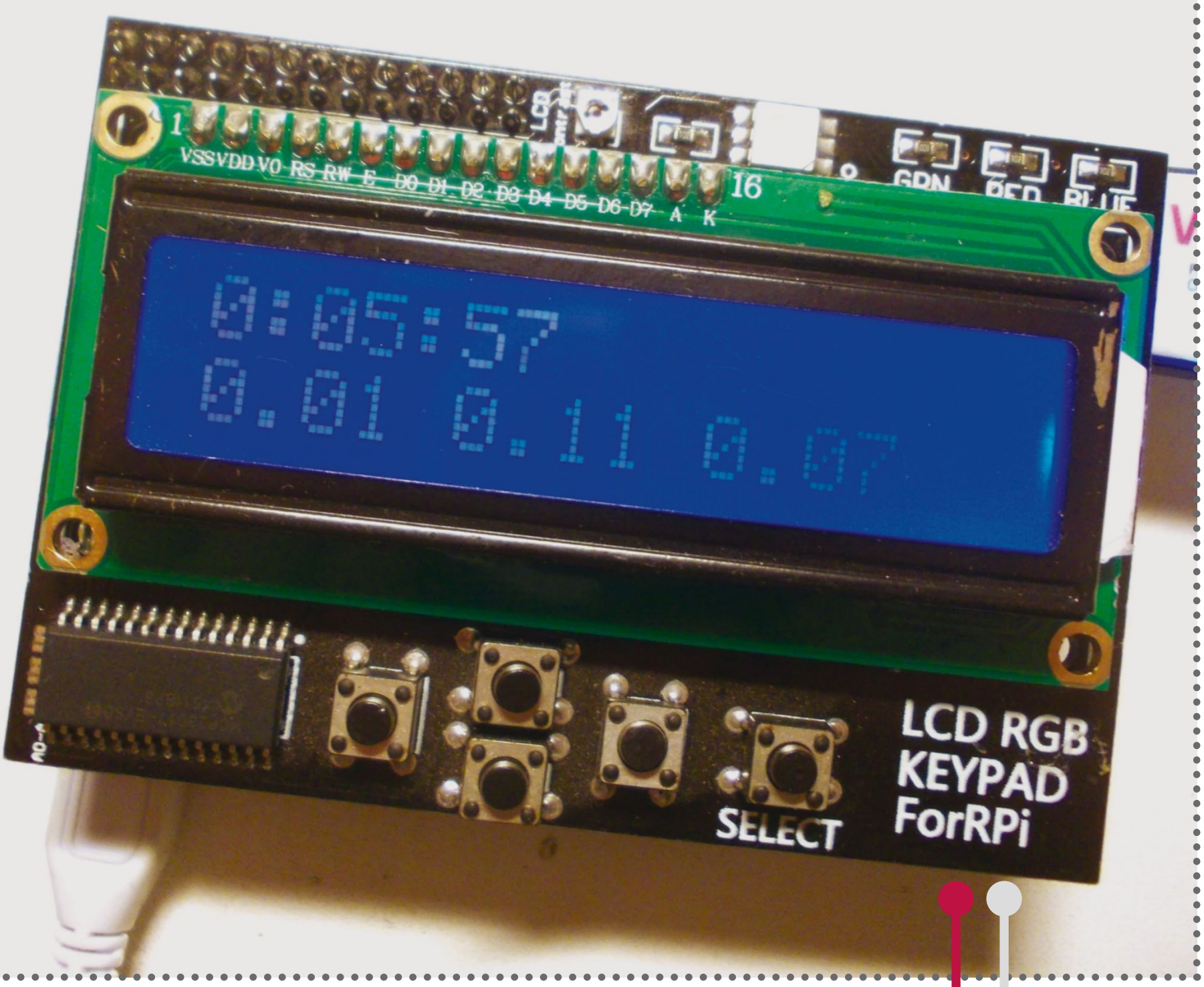
09 Power extreme!

More batteries added in parallel should result in almost double the uptime (at least 16 hours on a 256MB Raspberry Pi Model A), but instead of alkaline or rechargeable batteries you might consider a modern lithium-based AA cell, which will last considerably longer than alkaline batteries.



Control your Pi with pyLCI

Stop using all those weird console commands – this is an easier way to control your Pi



pyLCI is a simple hardware interface you can use on your Pi to control and configure it without the usual requirement of having a HDMI monitor, Ethernet connection or a connecting a USB-UART dongle.

Mainly, it enables you quickly configure the most important settings, such as connecting the Pi to a wired or wireless network and getting its IP address. This makes your Pi accessible – whether you are in the comfort of your home, somewhere outdoors, or maybe even working with your Pi on a long bus ride! All you need is a simple shield with a character display (16x2, for instance) and some buttons; there are many suitable shields which you can get, some of them as cheap as £5.

pyLCI stands for 'Python-based Linux Control Interface' and it's a user-friendly interface, too – you don't need to remember commands or spend time connecting things, and even people that are not technically literate can use it. In general, it saves you a lot of time when setting up another way to access your Pi (SSH, say), or monitoring it. It does not replace the command-line completely; rather, it helps you get through the initial hassle of needing to know the IP address which could otherwise waste a lot of your time, and it allows you to make your Pi projects considerably more user-friendly.

Where pyLCI is useful

There are places where pyLCI can be particularly useful – in educational workshops, for example. When you have multiple Pis on a table and students connect to them through SSH, you might need the IP

THE PROJECT ESSENTIALS

pyLCI Source

https://github.com/CRImier/pyLCI

pyLCI documentation

https://pylci.rtfd.io

address for "the Pi to the left of that guy"; with pyLCI, it takes five seconds.

Likewise, it helps on hackathons. If, at the beginning, you have to spend time searching for a HDMI-capable monitor or a cable, you'll have less time (and energy) to actually work on your idea. With pyLCI, you can concentrate on what's important, and you can even make the inevitable debugging easier by writing a simple pyLCI plugin for monitoring your software.

Also, it helps you transport your Pi projects safely. Say you want to bring your latest Pi-based project to work and show it to your co-workers – there's a chance that it'll stop working once you've transported it, and the more parts are involved in your project, the bigger that chance is. Whether the culprit is the lack of an internet connection, a loose wire, a hard-coded IP address or just a command failing to run on startup, you can easily connect to your Pi and figure out what the problem is.

One more kind of project with which pyLCI is helpful is Pi-based home servers. Whether you're running PiHole, a small webserver or using the Pi as a router, having a tool that lets you take a quick peek under the bonnet is useful, and it's quicker since you don't have to use SSH or access a web interface. You can check whether your Pi is actually running and the OS hasn't crashed, restart services, unmount mounted drives and run scripts – and you can also run shell commands from the interface (entering them is fairly slow, but there's a command history so you don't have to repeat yourself).

Most popular 'character display and buttons' Pi

Why Python?

It's the official language of the Raspberry Pi. Read the docs at python.org/doc

shields are supported out-of-the-box. For example, Adafruit's 16x2 Character LCD + Keypad for Raspberry Pi, which has a cool character display, with RGB backlight; you can get it from Adafruit (www.adafruit.com) or one of its distributors. There's also Piface Control and Display 2 with three more buttons and an IR receiver, available from Element14 (www.element14.com). There are also the no-name LCD RGB KEYPAD ForRPi shields, which are clones of the Adafruit shield with minor hardware changes (and a separate RGB LED instead of RGB LCD backlight), available on eBay and similar sites.

If you've already got a different shield which does have an LCD and buttons, it's likely that it can work with pyLCI; you'll just need to enter the GPIOs used by the display and the screen into a config file (see OTHER_SHIELDS.md in the pyLCI source for instructions)

As for the software, you can install pyLCI on any Pi that runs Raspbian Jessie or later. Check whether it's compatible by running:

```
cat /etc/os-release
```

If your VERSION_ID is 8 or more, you can install pyLCI; if not, a system upgrade might be in order!

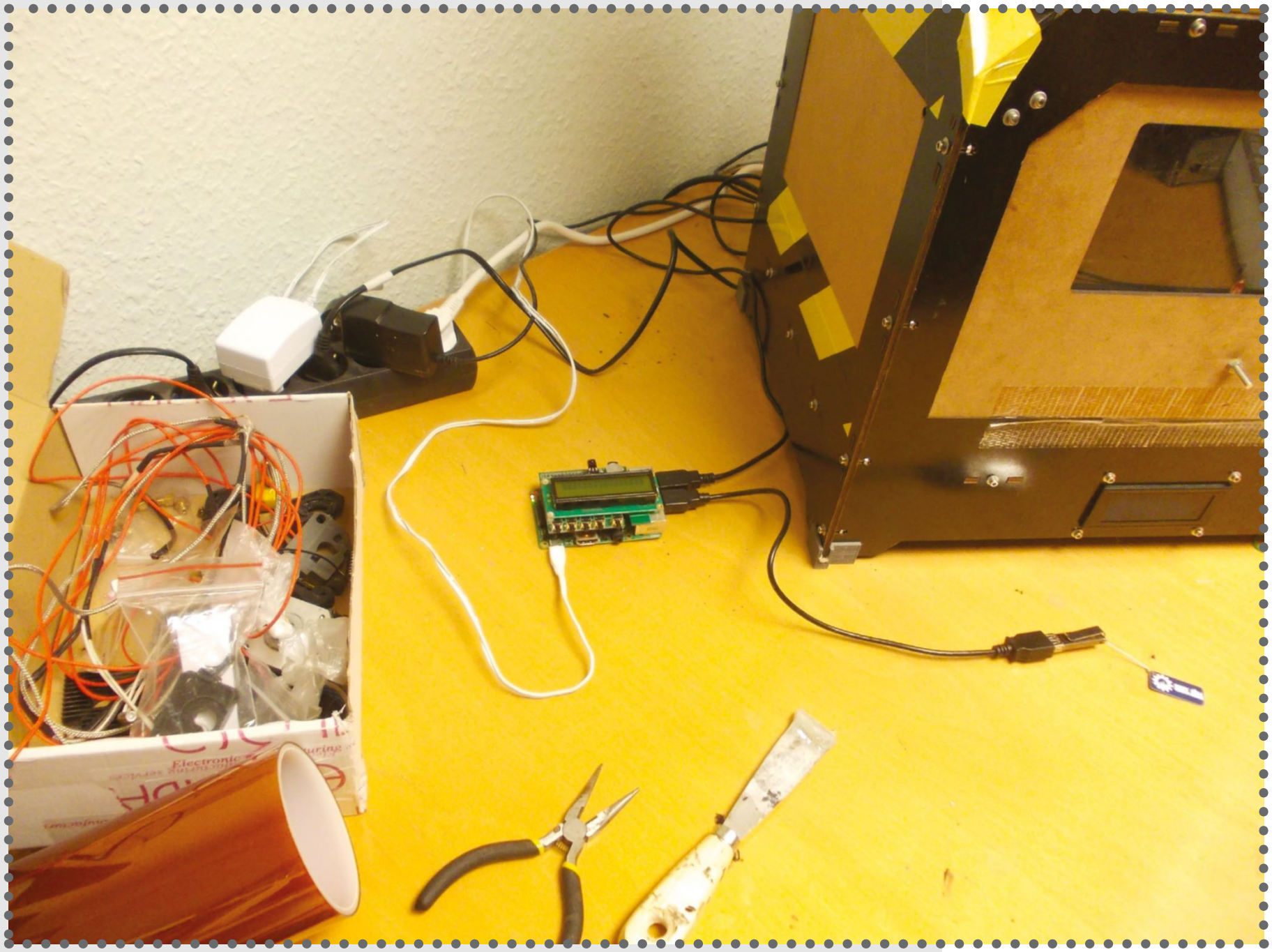
Installation and usage

To install pyLCI, download it from GitHub, install its dependencies, run the configuration script and pick your shield – pyLCI will configure itself for that shield. Then run pyLCI manually to make sure your hardware

works. Once you've confirmed that it does, sync your pyLCI code to the global pyLCI installation (which runs on system startup). The sync mechanism is there so that your local changes won't break pyLCI until you sync. Here's some code to do all these things:

```
apt-get install git
git clone https://github.com/CRImier/pyLCI
./setup.sh
./config.sh
sudo python main.py
./update.sh
```

Your screen should light up, and "Welcome to pyLCI"



Below A Pi using Octoprint - pyLCI helps with webcam and flash drive operations

should appear on it. Right after, you'll see a menu: use the Up and Down buttons to navigate, and Enter to go into submenus. To go back, use the Left key. The Right key will occasionally be used for context menus or character input. PiFace Control and Display has the Left and Right as two leftmost keys in the bottom row, with the other keys unused.

Capabilities

You can easily monitor your uptime and CPU load from pyLCI; go to System > System info > Uptime&load. It's handy for monitoring your Raspberry Pi creations, when you can glance at the display and see if your Pi is okay and hasn't rebooted when you didn't expect it to. It is even more useful if you end up compiling something on a Pi, or, say, streaming video from it – you can check the CPU load and see if the compiling process that you've launched has finished, which is very convenient because you no longer have to check the terminal all the time.

Additionally, if your HAT has a RGB screen backlight (like the Adafruit HAT) or an RGB LED of some sorts, the CPU monitor will make use of it, lighting it red when CPU load is nearing 100%, green when it's below 30%, and blue/violet for values in between – making for an even more convenient monitoring tool!

Most importantly, you can connect to Wi-Fi using pyLCI, by going to Networking > Wireless. If a wireless card is found, you'll see a menu of possible actions. Pick Scan to make the Wi-Fi card re-scan the networks; wait for five seconds, then go to Networks to see the networks that are available. Pick the

"You can easily monitor your uptime and CPU load from pyLCI"

network you want to connect to; if it has a password set, it will request you to enter the password. You can do this using the arrow keys: use Up or Down to change the character you're on, and Left or Right to change the character itself.

Entering a password this way is tiring, but thankfully, you shouldn't have to do it twice! To make it even quicker, move up for letters (both lowercase and uppercase) and down for numbers and special characters. If you mess it up and enter too many characters, you can also clear your last character using backspace (move down once to see it). You can also check your IP address. Go to Networking > Interfaces and select your interface (for example, eth0). Scroll down a little bit to see the IP address (if you received one, that is).

You can also run scripts and commands from pyLCI. This enables you to automate parts of your life; for example, you might have a flash drive with important information that you don't want to lose, and you want to regularly back it up on some kind of device. You can write a script that detects your device, mounts it and uses rsync to synchronize the flash drive's contents. Then, when you come home to your Pi, go to Scripts > Script by path, find your script on the Pi and click Enter on it – it'll run your script (and you can even give it command-line arguments before running).

"Most popular 'character display and buttons' Pi shields are supported out-of-the-box by pyCLI"

An example app

All the functionality in pyLCI is provided by Python 'apps'. It's easy to write one and add it to your pyLCI install. Here's a pyLCI 'Hello world':

```
# Caption: main.py
  menu_name = “Hello world”
  from ui import Printer
  i = None # An input device object
  o = None # An output device object
  def callback(): # This will be called when
    the app is selected
       Printer(“Hello, world!”, i, o, 5)
```

This code prints “Hello, world!” on screen for five seconds once the “Hello world” app is selected in the main menu. Simply saving this file somewhere is not enough – you have to store it inside pyLCI. Go to the directory where you downloaded pyLCI into (most likely /home/pi/pyLCI). All apps are stored in the apps folder; create a new folder inside apps and name it something like hello_world, then place the main.py file inside that folder. Also create a new empty file in that folder, named __init__.py. Here are the commands you can use to do this, assuming you downloaded pyLCI into /home/pi/pyLCI:

```
  cd /home/pi/pyLCI
  mkdir apps/hello_world
  nano apps/hello_world/main.py # Paste the
    code there
  touch apps/conncheck/__init__.py
```

The last command opens an editor, pastes your code into it and exits. That’s all you need to create your own pyLCI app. To test it, you can run pyLCI manually. First, run sudo systemctl stop pylci to stop the global pyLCI

process (so that it doesn't try to grab the screen and buttons you're using). Then run sudo python main.py, use the buttons to scroll down to your application and press the Enter key. "Hello, world!" should appear on the screen.

Writing your own app

Of course, an app is only truly useful once you can somehow interact with the outside world. Let's take a real-world situation: there's something wrong with your network and you want to know where the problem lies – is it your computer, your router or your ISP? Let's write a simple pyLCI app for that.

First, what do we check for? There are four typical places for connectivity problems to appear: the computer you're using, your router, the DNS or your internet provider. To cover most of those, we can: 1) check whether we can ping an IP address that belongs to Google, ruling out ISP problems; 2) try to ping a website, referring to it by its domain name – making sure our problem is not a DNS problem.

There's a great Python library that lets us do both of these tasks easily – it's called pyping. Let's install it:

```
sudo pip install pyping
```

Note that due to the way ping works on Linux, you need to use the pyping library as root – that is, using sudo. As pyLCI has to run as root anyway, this is not a problem for our app, but it's something to remember while experimenting.

The pyping module has a ping function, which accepts a string: either an IP address or a domain

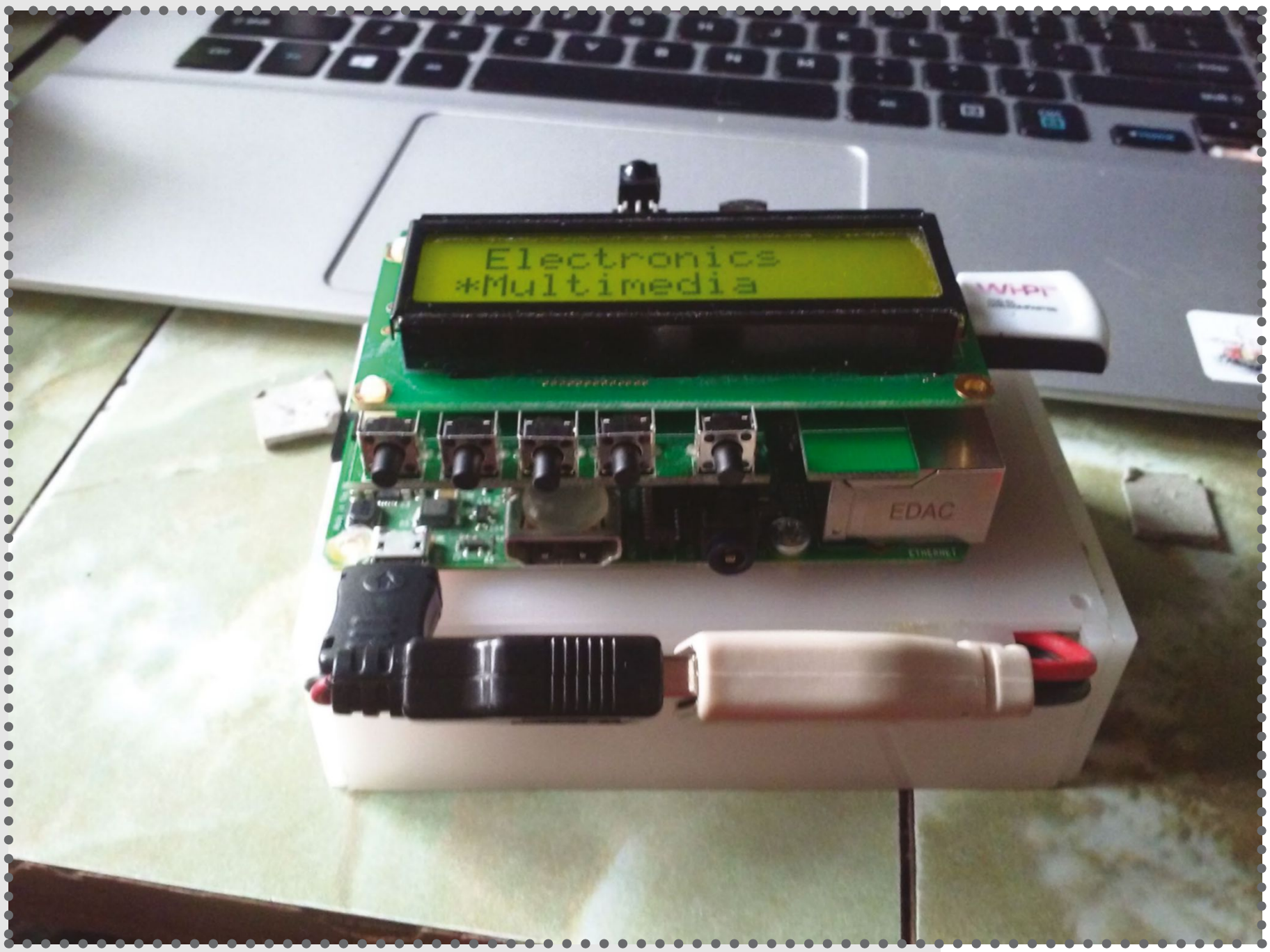
Catch-22 banished

pyLCI solves the chicken-and-egg problem of connecting a Pi to a Wi-Fi network, when you need to use SSH to enter the Wi-Fi password, but you can't use SSH until you're connected to a network. Using just the screen and five buttons, you can pick a network and enter its password, then see the IP address that you've received from the router.

name. It raises an exception if the name can't be resolved – and if we're trying to resolve a well-known name such as google.com, this would likely mean that DNS isn't working for us. Otherwise, it returns a response object, with a ret_code property; if the destination was reached, it's set to 0, otherwise it's set to 1. Perfect! So let's write some simple code that uses this to check whether an internet connection is accessible. First, let's check the internet connection in general, then whether the DNS works:

```
menu_name = "Connection test"

```



Below A portable battery-powered Raspberry Pi, with pyLCI greatly improving the portability

```
import pyping # Our library of choice
from ui import Printer

i = None; o = None

def callback():
    success = pyping.ping("8.8.8.8").ret_code
    if success == 0:
        Printer("Connection working!", i, o)
    else:
        Printer("Connection failed!", i, o)
    # Then, check DNS
    try:
        pyping.ping("google.com")
    except:
        Printer("DNS failed!", i, o)
    else:
        Printer("DNS working!", i, o)
```

This is all we need – our proof-of-concept app is done. Let's put it with the other networking-related apps in pyLCI, in the apps/network_apps directory:

```
mkdir apps/network_apps/conncheck
nano apps/network_apps/conncheck/main.py
# Again, paste our code in the editor
touch apps/hello_world/__init__.py
```

Adding features

There's one more important feature we can add to our new app: 'captive portal' detection. When you connect to certain Wi-Fi hotspots, such as those in coffee shops, they won't let you access the internet

"In more complicated apps, you can use RPC so that your app code isn't contained in pyLCI"

straight away; instead, they redirect you to a page that requires you to agree with their Terms of Service, or even create a login and password.

This means that your Pi won't be able to access the internet straight away and, in many cases, will also mean that you won't be able to SSH into it. Now, we can't yet do much in these kind of situations, but we can at least detect it, so that you spend less time on debugging your network connection.

To detect if a captive portal is redirecting us somewhere else, we can fetch a known page and check its contents. One example of such a known page is IcanHazIP, http://icanhazip.com. If you visit this page, you'll see your external IP on a blank page, and that's it. So we can trivially check if we're getting an IP address, or if we're being redirected to somewhere else – and we can get our external IP address, too. At the top of main.py, with all the other imports, let's import the requests library – we can use it to download the webpage:

```
import requests
```

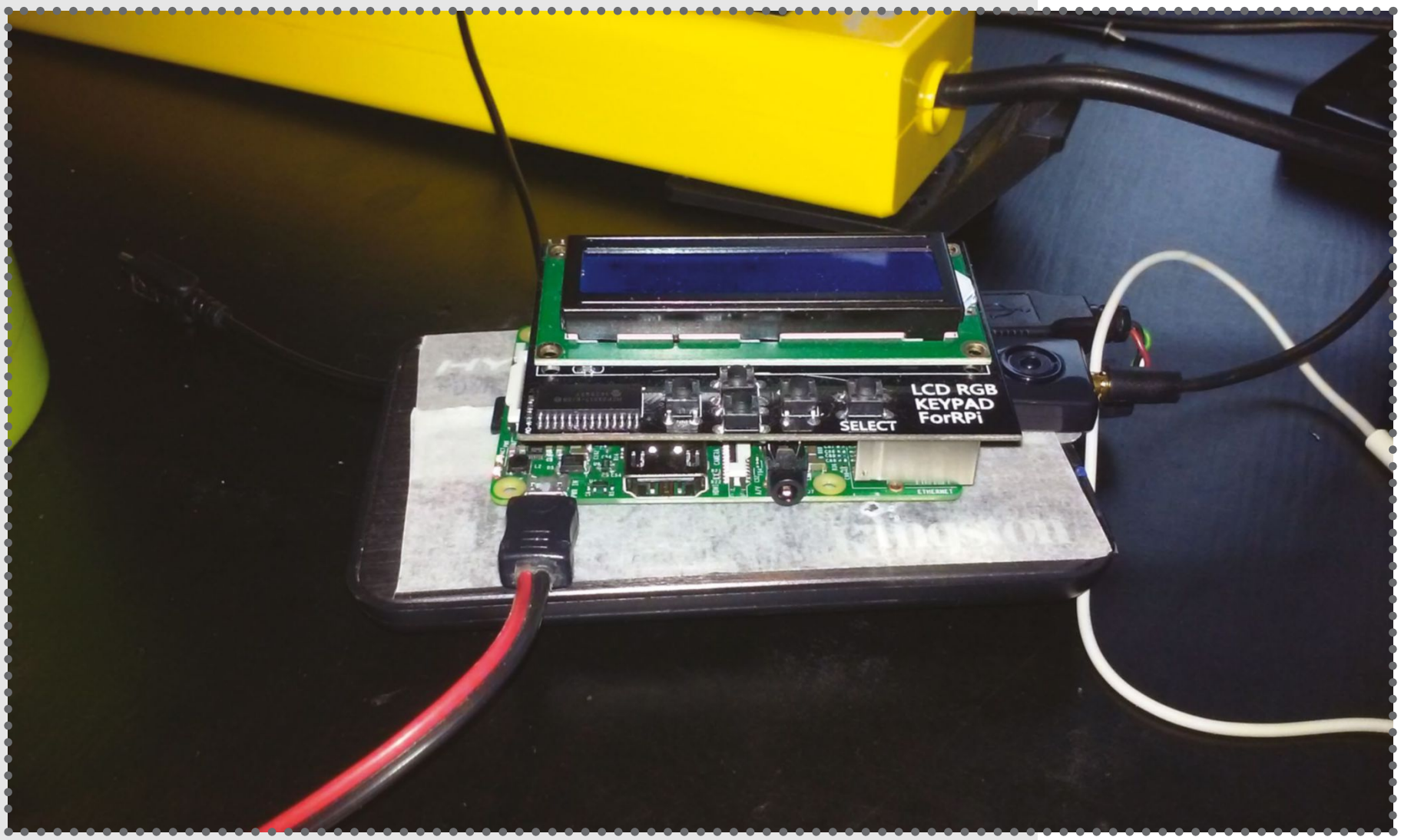
Then let's add a function for captive portal check:

```
def is_captive_portal():
    try:
        r = requests.get("http://icanhazip.
com")
        assert (r.status_code == 200)
```

At this point, if the status code is not 200 we've been redirected somewhere – which shouldn't happen

with this website. So if the code is not 200, an AssertionError will be raised and we'll go straight to the except AssertionError block. However, that might not be enough; let's parse the text we received from the page, to make sure that what got is actually an IP address – in other words, it's four numbers separated by three dots:

```
            ip = r.text.strip()
            assert (ip.count(".") == 3)
            parts = ip.split(".")
            assert all([part.isdigit() for part
    in parts])
        except requests.ConnectionError:
            # Internet connection problem?
            return False
        except AssertionError:
            return True
```



Below A spectrum monitoring station with an RTL-SDR dongle and an SSD; pyLCI shows system status

```
        else:
            return False
```

In the callback, let's add one more stage of checks:

```
    [...]
            Printer("DNS working!", i, o, 3)
    if is_captive_portal() == True:
    Printer("Captive portal detected!", i, o)
    else:
    Printer("No captive portal found!", i, o)
```

As you can see, it's easy to add your own functions to pyLCI. In more complicated apps, you can use RPC so that your app code isn't contained in pyLCI, and pyLCI has helpers functions for this, too. Happy hacking!

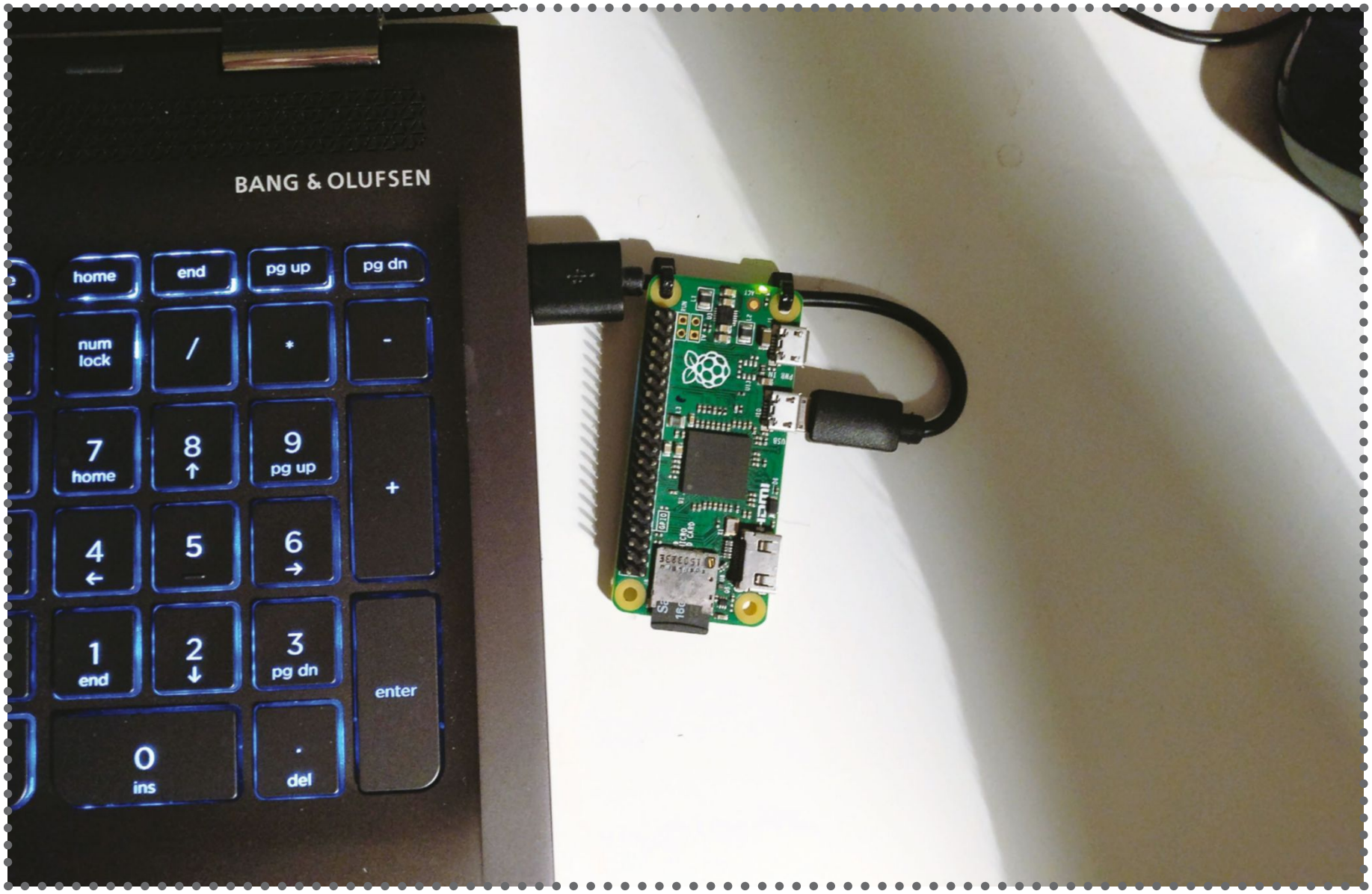
pyLCI can save you

If you ever were in a situation where you couldn't connect to your Raspberry Pi, it's likely pyLCI could have solved that problem. You do need to install it beforehand, but you only need to do it once - after installing, it will be there for you, waiting to save your day…

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

"Access a Raspberry Pi Zero from a laptop"



Get this issue's source code at:

www.linuxuser.co.uk/raspicode